A Transformation Approach of Mobile Activity Diagram Towards Nested Petri Net

Guerrouf Faycal, Allaoua Chaoui, Al-dahoud Ali

***Abstract*—UML is the de facto standard language for modeling object oriented software systems. Mobile activity diagram is an extension of the standard UML activity diagram that deals with the requirement to model, specify, and visualize mobile agent based systems. However, it inherits its lack of formal notation for analysis and verification purpose. In this paper we propose an approach for transforming mobile activity diagrams to nested petri nets models. The meta-modeling tool AToM3 is used to create meta-models for mobile activity diagram and nested petri net. A graph grammar is proposed for automatic transformation. An auctioning system example illustrates our approach.**

***Index Terms*—UML, Mobile Activity Diagram, Nested Petri Net, Meta-modelling, Graph Grammars, Graph Transformations, AToM3.**

I. Introduction

MOBILE agents are autonomous entity. They can move from a host to other hosts, and communicate together in a network of heterogeneous computer systems [1]. They are used in many domain such us electronic commerce, information retrieval, remote device monitoring, workflow applications and groupware [1], [2]. There special features had made the need to a new modeling tools and techniques.

The Unified Modeling Language (UML) [3] is considered nowadays as a standard software modeling language [4]. It includes thirteen diagram categorized in structural and behavioral diagrams. The first set are used to model the structure of a system, and the second set are used to model the dynamic of a system [3]. Activity diagram is one of the five diagrams to model the dynamic aspects of systems. The activity diagram is typically used to describe the activities performed in a general process workflow, though it can also be used to describe other activity flows, such as a use case or a detailed control flow [5]. Mobile UML activity diagram is an extension of the standard UML activity diagram that deals with the requirement to model, specify, and visualize mobile agent based system. The extension is specified in UML itself, using the UML extension mechanism like stereotypes and tagged values. Therefore, mobile activity diagram inherits the lack of UML in formal notation for analysis and verification purpose.

Petri nets are formal and graphical modeling language [6]. They are a very powerful tool for analysis and verification. Nested petri net [7] extend petri net to model mobile agents. They are a height level petri net in which tokens can be Petri nets themselves.

The perfect modeling tool is the one that combine both the intuitive and graphical notations of mobile activity diagram and the formal power of nested petri net. The modeler uses the mobile activity diagram formalism to model the system which is automatically analyzed and verified in a transparent way using its equivalent model in nested petri net formalism.

In this paper, we propose a graph transformation approach and tools based on the combined use of meta-modeling and graph transformation grammars. First, we propose a meta-model for mobile activity diagram, and nested petri net. Then the $AToM^3$ tool is used to generate a visual modeling tool to process models in mobile activity diagram and nested petri net formalism. Finally, we propose a graph grammar to perform the transformation.

This paper is organized as follows: in section II we overview some related works. in section III We review the most relevant notions to our approach. in section IV we describe our approach, we start by defining the meta-models for mobile activity diagram and nested petri net, those two last are used by $AToM^3$ [8] to generate a tools. then we propose a graph grammar to perform the transformation. in section V we present a case study to illustrate our approach. in section VI we concludes the paper .

II. Related Work

$AToM^3$ [8] is a powerful tool combining the meta-modeling, multi-paradigm and graph transformation. Many works had been realized using $AToM^3$. They are similar from the point of view of the process pursued to achieve the transformation. In which, a meta-model for the source graph being transformed and a meta-model for the destination graph are defined. Given these meta-models, tools are generated using $AToM^3$. Then a graph grammar is described in term of source and destination models to perform the transformation.

In [9], the authors have presented an approach that automatically generates a Maude specification from ECATNets [10] models. an ECATNets meta-model had been proposed with the meta-modeling tool $AToM^3$, to automatically generate a visual modeling tool to process models in ECATNets formalism, then a graph grammar had been defined to translate the models created in the generated tool to a Maude specification. In [11] the authors proposed a graph grammar to transform non-deterministic finite state automata (NFA) to their equivalent deterministic finite state automata (DFA). In [12] the authors

Guerrouf Faycal is with the Department of Computer Science, Faculty of Engineering, University El Hadj Lakhdar Batna, Algeria, e-mail: fguerrouf@yahoo.fr.

Allaoua Chaoui is with the Department of Computer Science, Faculty of Engineering, University Mentouri Constantine, Algeria, e-mail: a_chaoui2001@yahoo.com.

Al-dahoud Ali is with the Department of Computer Science, Faculty of Science and IT, AL-Zaytoonah University of Jordan, Jordan, e-mail: aldahoud@alzaytoonah.edu.jo.

proposed a transformation of sequence diagrams, to statecharts diagram. In [13] the authors had defined meta-models for both syntax and semantics of Statecharts (without hierarchy) and petri nets. The authors also proposed a graph grammar for the transformation between statecharts and petri nets. Other works could be found in [8].

III. Background

This section briefly introduces the mobile activity diagram, nested petri net, and $ATоM^3$

A. Mobile Activity Diagram

Activity diagram allow to specify how the system will accomplish its goals. It show high-level actions chained together to represent a process occurring in the modeled system [14]. Mobile activity diagram [15] is an extension to ordinary activity diagram used to model the dynamic behavior of mobile agents. Specific features of mobile agents, which model, are mobility, cloning, messaging between agents. The mobile activity diagram basically uses activity partition to model these features [15]. An activity partition is a kind of activity group for identifying actions that have some characteristic in common [3]. It may be multidimensional hierarchical partitions. A vertical dimension with the stereotype <<Host>> and parameter as its unique name (address) represents a location and another orthogonal dimension with the stereotype <<agent>> represents an agent. The mobility of agents between location is represented by a particular activity which is "Go" action [15]. The Figure 1 show an example of multidimensional partition modeling an agent "Agent1", which moves from location "host1" to "host2" by using the "Go" action.

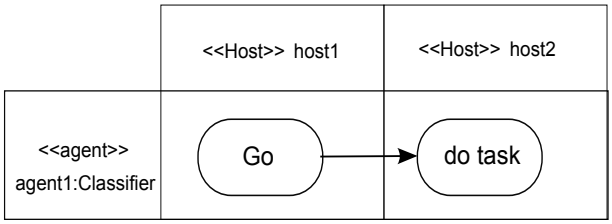


Figure 1. "Go" action

The cloning feature is modeled by an invocation activity as shown in Figure 2 This activity allows an agent to clone it first, then to send the cloned agent to another host. For lack of space, the reader is referred to [15]

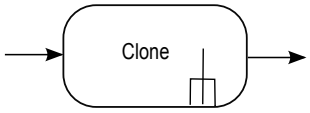


Figure 2. Clone action

B. Nested Petri Nets

Nested petri nets [16] are a high level petri net models that are convenient for modeling hierarchical multi agent systems. Element (token) in nested petri net may be a petri net witch may have also a petri nets as their tokens. Therefore the number of levels in nested petri net is not limited. A two level nested Petri net consists of a set of ordinary Petri nets defining the structure of a net tokens and a system net. A system net is a high level predicate transition net with a limited arc expression language and without transition guards [7]. The behavior of nested petri net is described by four kinds of steps [7], [16].

- *A transfer step* is a step in a system net which can "move" ,"generate", or "remove" its elements, but doesn't change their inner states.
- *An element autonomous step* affects only the inner state in one element.
- *Horizontal synchronization step* is the simultaneous firing of two element nets, located in the same place of a system net.
- *Vertical synchronization step* is the simultaneous firing of a system net together with its elements, involved in this firing.

Figure 3 shows an example of a nested petri net.

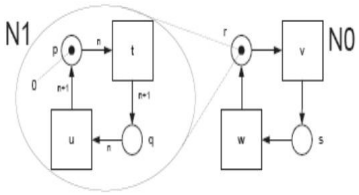


Figure 3. Example of a nested petri net

C. Graph Grammar and AToM3

$ATоM^3$ (A Tool for Multi-Formalism and Meta-Modeling) [8]. It was developed at the modeling, simulation and design Lab in the School of Computer Science of McGill University. It is written in python [17]. Its main purpose is the meta-modeling and model transformation [8]. In $ATоM^3$, formalisms and models are described as graphs. From a meta-model, $ATоM^3$ generates a tool to process (create and edit) models described in the specified formalism [11]. Graph transformation consists of applying a rule to a graph and iterating this process. Each rule has graphs in their left and right hand sides (LHS and RHS). In order to apply a rule to a graph (called host graph) a matching has to be found between the LHS of the rule and a part of the host graph. If such a matching is found, the elements in the host graph can be substituted by the elements in the RHS [18]. Rules can have applicability condition, as well as action to be performed when the rule is applied. In $ATоM^3$ rules are ordered according to a user-assigned priority, and are checked from higher to lower priority [19].

IV. Our Approach

The transformation is performed between a source graph which is the mobile activity diagram and a destination graph which is the nested petri net formalism. To reach this goal we pursue the following steps:

1) Define the meta-model for mobile activity diagram, then use $AToM^3$ to generate a tool for our given meta-model.
2) Repeat the same step for nested petri net.
3) Define the graph grammar to perform the transformation.

A. Meta-model of Mobile Activity Diagram

The UML document specification has defined many different levels of activities: i) fundamental ii) basic iii) intermediate iv) complete v) structured vi) complete structured vii) extra structured. each level adds its own constructs addressing a particular area [3]. The most suited to convert into Petri net models are fundamental, basic and intermediate activities [20]. To define our meta-model, we had combined these three levels and took in consideration The new stereotype introduced by the extension. Our proposed meta-model for mobile activity diagram consists of 09 classes and 08 associations as shown in Figure 4

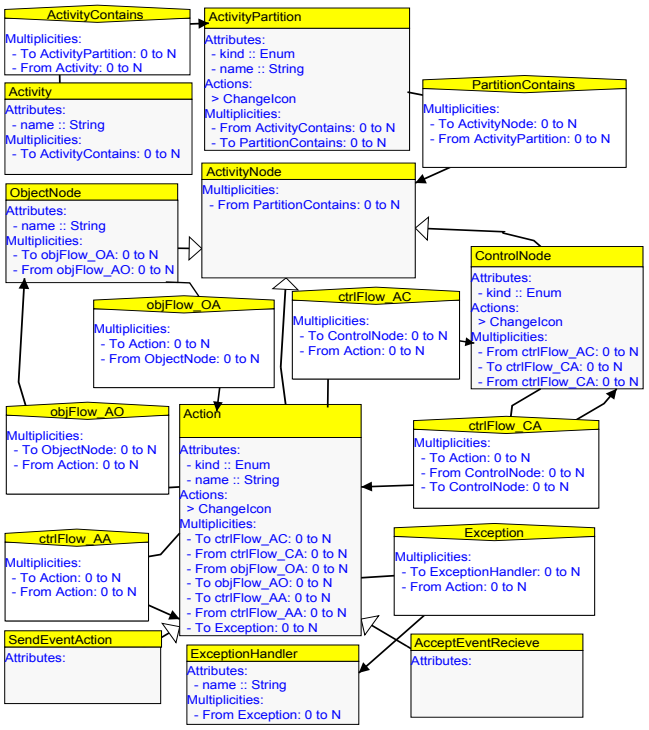


Figure 4. Meta-model of mobile activity diagram

We had defined a graphical appearance for the classes according to their standard specification in [3] and the definition of the extension in [15].

Given our meta-model, we have used $AToM^3$ to generate a modeling tool. It can be used to process models in the mobile activity diagram formalism. The generated tool is illustrated with an example in Figure 5.

B. Meta-model of Nested Petri Nets

Our meta-model consists of 7 classes and 8 associations as shown in Figure 6. It is mainly formed of classes that define the element nets and others define the system nets. The class "EN" represents an element net. It is an ordinary petri net, the classes "EPlace", "ETransition" are their places and transition

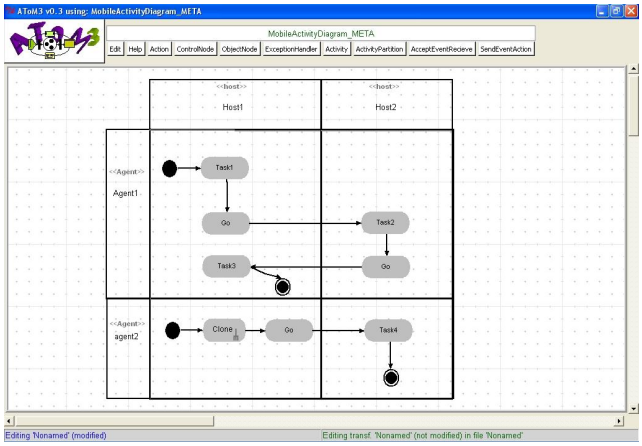


Figure 5. Generated tool for mobile activity diagram

respectively. The class "SN" represents the system net, the classes "SPlace", "STransition" are their places and transition respectively.

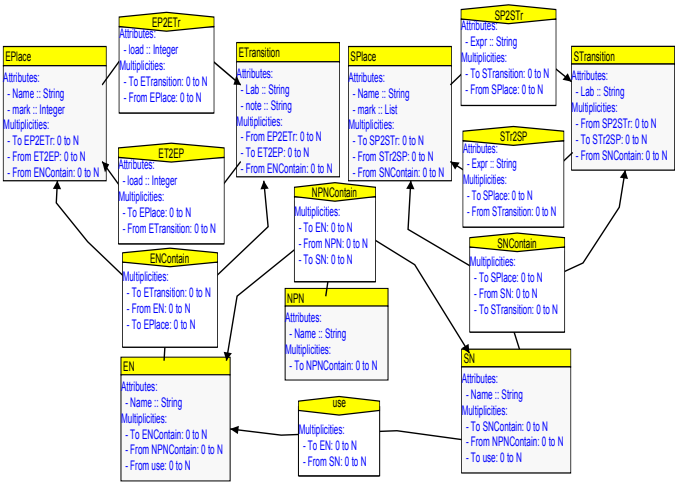


Figure 6. Meta-model of nested petri net

Based on this meta-model, $AToM^3$ generates a tool to model the nested petri net graphically as illustrated in Figure 7.

C. Graph Grammar: Converting Mobile Activity Diagrams into Nested Petri Nets models

Our graph grammar is composed of sixty one rules categorized in 04 categories:

- Rules to create the system net.
- Rules to synchronize between the element nets and the system net.
- Rules to create the element nets.
- Rules for cleaning the result of transformation.

The idea of the transformation in our grammar is that:

- Every activity partition of kind host represents a place in the system net, and every activity partition of kind agent represent an element net.
- Every mobile action in the mobile activity diagram represent a vertical synchronization in the nested petri net,

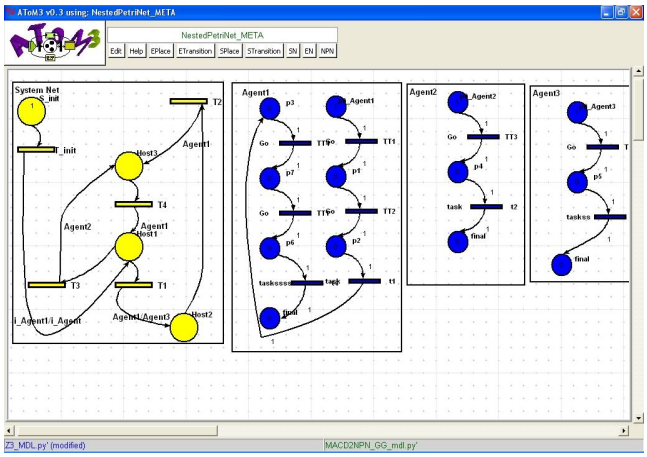


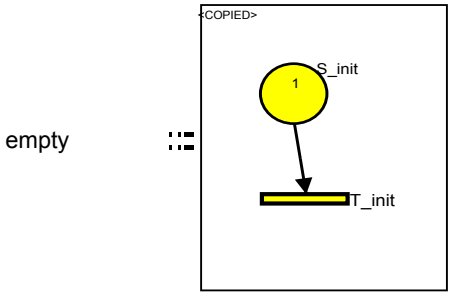
Figure 7. Generated tool for nested petri net

because it does move an agent from one host represented by a place in the system net to other host represented by another place in the system net.

- Every clone action represent a transfer step in the nested petri net.

We were inspired by [20] to map the rest of the entities in the mobile activity diagram to the nested petri net formalism.

1) rule1 SystemNetCreate (priority 1) : This is the first rule applied in the grammar. It creates the system net of the nested petri net with a default name "system net". This partial system net is formed of one initial place and transition; their names are respectively "S_init", "T_init" (Figure 8). The place has one token of type black dot because it is considered as a current state. For each nested petri net, there's a single system net. This constraint is interpreted by a condition to check if the rule is not executed before.

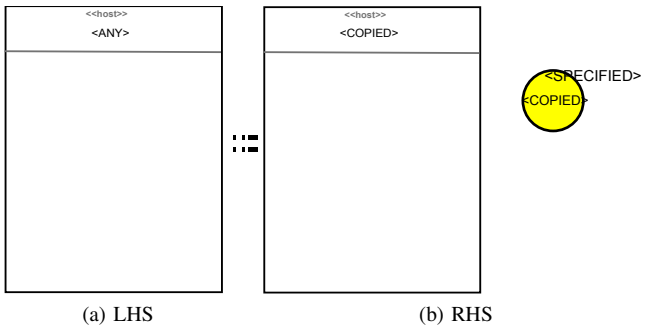


(a) LHS (b) RHS

Figure 8. rule1 SystemNetCreate

2) rule2 Host2SPlace (priority 2): This rule allows us to transform a simple activity partition of kind "Host" to a place in the system net and has the name of this activity partition (Figure 9). Each place represents a possible location in which the agents could evolve. The number of tokens on all places transformed is 0 because they are not considered as a current state.

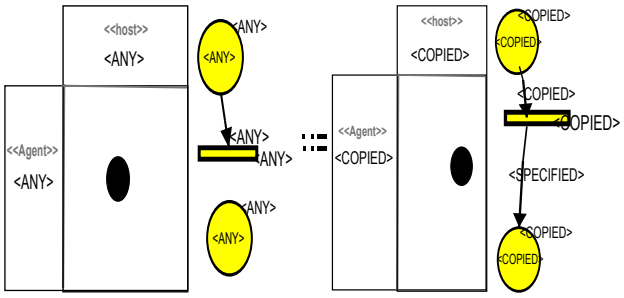
3) rule3 SPlace2Init_1 (priority 3) : The initial transition represents the starting point of the system in which no element net (agent) does exist yet in the system. The firing of this



(a) LHS (b) RHS

Figure 9. rule2 Host2SPlace

transition creates the entire element net that could exist at the first run of the system and puts them into their places. Therefor we apply this rule (Figure 10) to identify the places of the system net that should be connected with the initial transition. Each activity partition of kind "host" containing an initial activity node corresponds to a place in the system net that meets the condition for being connected with the initial transition. It creates an arc between the identified place and the initial transition. The expression of the created arc will be the concatenation of the letters "i_" and the name of the element net.



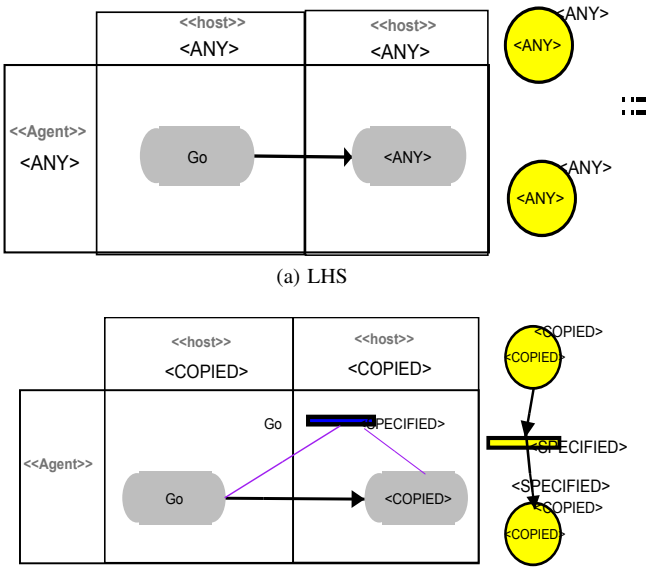
(a) LHS (b) RHS

Figure 10. rule3 SPlace2Init_1

4) rule6 Go2VerticalSync_2 (priority 6) : Every mobile action (Go action) is mapped into a vertical synchronization in the nested petri net. Thus this rule (Figure 11) allows us to:

- Search for "Go" action and create a transition with the name "Go" in the element net involved in the synchronization.
- Create a transition between the places in the system net corresponding to the source and destination of the mobile action. The expression arc created in the system net will carry the name of the agent had performed the "Go" action.
- Give an adjacent name to the tow transition so they will be synchronized vertically.

5) rule9 ElementNetCreate (priority 9) : This rule (Figure 12) allows us to transform an activity partition of kind "agent" on the mobile activity diagram to an element net. The name of the created element net is the same name of the agent. This Rule allows us also to transform the initial activity node contained in that activity partition to the initial

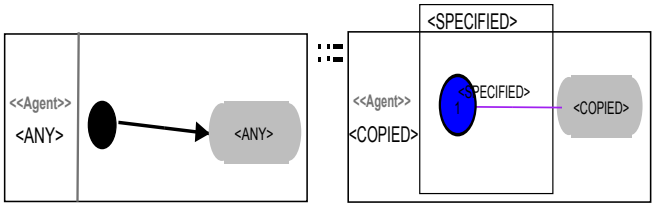


(a) LHS

(b) RHS

Figure 11. rule6 Go2VerticalSync_2

place of the created element net. The name of the initial place is the concatenation of the letters "p0_" and the name of the agent. The number of the tokens is 1 because it is considered as a current state. The element nets created are used by the system net as tokens.



(a) LHS (b) RHS

Figure 12. rule5 ElementNetCreate

V. Case Study:

In order to demonstrate our approach we will use the example presented in [15]. This case study consists of two agents; an Auctioneer agent, who is stationary, and a bidder mobile agent. The auctioneer agent resides at seller's host. The Bidder mobile agent will circulate round potential bidders while gathering bids.

For the lack of space and clarity of the example we will omit the ordinary actions and control and keep the rest. The Figure 13 illustrate the mobile activity diagram model of the example.

After applying the grammar, we have obtained the final nested petri net shown in Figure 14. We can see that we have two elements net that represent the auctioneer agent and the bidder agent. We have also four places that represent the different location in the modeled system. The auctioneer

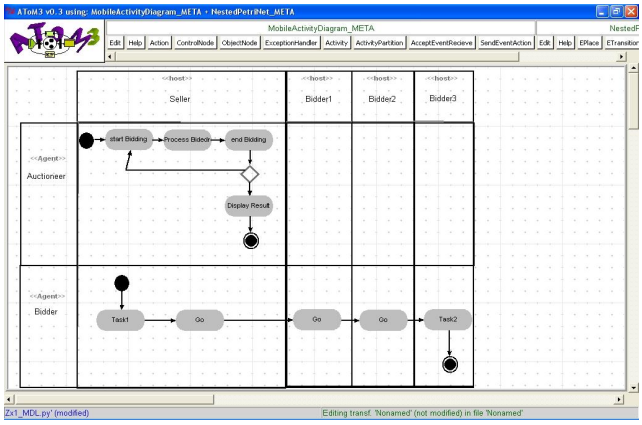


Figure 13. Auction system model

agent has no adjacent labels with the system net because he is a stationary agent, unlike the bidder agent which has an adjacent labels with the system net. For example of vertical synchronization we have the adjacent labels T2 in the system net, and TT2 in the element net (bidder) because the bidder move from the host "bidder1" to the host "bidder2".

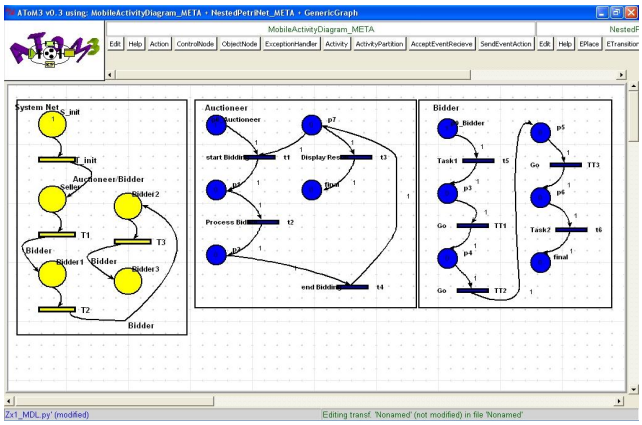


Figure 14. The resulting nested petri net model

VI. Conclusions

In this paper we have proposed an approach based on combining meta-modeling and graph grammars for transforming mobile activity diagrams into nested petri nets models. The approach tends to take the advantage of the mobile activity diagram and cover its lack of formal notation by offering an equivalent nested petri net for analysis and verification purpose.

We have used $ATOM^3$ to create meta-models for both mobile activity diagram and nested petri net, from which $ATOM^3$ generated tools. The mobile activity diagram tool could be used to create model for a given mobile agent based system, and be transformed into nested petri net using our graph grammar created also with $ATOM^3$. Our approach has been illustrated by an example.

REFERENCES

[1] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding code mobility," *IEEE Transactions on Software Engineering*, vol. 24, no. 5, pp. 342–361, Feb. 03 1998. [Online]. Available: http://citeseer.ist.psu.edu/182286.html;http://www.cs.ucsb.edu/~vigna/pub/fuggetta_picco_vigna_understanding.ps.gz

[2] M. Oshima and D. B. Lange, "Seven good reasons for mobile agents," *Communications of the ACM*, vol. 42, March 1999.

[3] O. M. Group, "Unified modeling language superstructure," 2004, version 2.0. [Online]. Available: http://www.omg.org/cgi-bin/doc?ptc/2004-10-02

[4] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.

[5] H.-E. Eriksson, M. Penker, and D. Fado, *UML 2 Toolkit*. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[6] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, April 1989.

[7] I. A. Lomazova, "Nested petri nets – a formalism for specification and verification of multi-agent distributed systems," *FUNDINF: Fundamenta Informatica*, vol. 43, pp. 195–214, 2000.

[8] "Atom3 home page," 17 January 2008. [Online]. Available: http://atom3.cs.mcgill.ca/

[9] E. Kerkouche and A. Chaoui., "A graphical tool support to process and simulate ecatnets models based on meta-modelling and graph grammars." *INFOCOMP Journal of Computer Science*, vol. 8, no. 4, pp. 37–44, 2009.

[10] M. B. M. Maouche, M. Soualmi, and M. Boukebeche, "Protocol specification using ecatnets," *Networking and Distributed Computing*, pp. 7–35, 1993.

[11] J. de Lara and H. Vangheluwe, "AToM^3: A tool for multi-formalism and meta-modelling," in *FASE*, ser. Lecture Notes in Computer Science, R.-D. Kutsche and H. Weber, Eds., vol. 2306. Springer, 2002, pp. 174–188. [Online]. Available: http://link.springer.de/link/service/series/0558/bibs/2306/23060174.htm

[12] X. Sun and H. Vangheluwe, "A model-driven approach to scenario-based requirements engineering," 2007.

[13] J. de Lara and H. Vangheluwe, "Computer aided multi-paradigm modelling to process petri-nets and statecharts," in *ICGT*, ser. Lecture Notes in Computer Science, A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, Eds., vol. 2505. Springer, 2002, pp. 239–253. [Online]. Available: http://link.springer.de/link/service/series/0558/bibs/2505/25050239.htm

[14] K. Hamilton and R. Miles, *Learning UML 2.0*. O'Reilly, April 2006.

[15] M. Kang, L. Wang, and K. Taguchi, "Modelling mobile agent applications in UML2.0 activity diagrams," Apr. 21 2004. [Online]. Available: http://citeseer.ist.psu.edu/668251.html;http://www.auml.org/auml/supplements/UML2-AD.pdf

[16] I. A. Lomazova, "Nested petri nets: Multi-level and recursive systems," *Fundam. Inform*, vol. 47, no. 3-4, pp. 283–293, 2001.

[17] "Python programming language." [Online]. Available: http://www.python.org/

[18] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Plump, A. Schürr, and G. Taentzer, "Graph transformation for specification and programming," *Science of Computer Programming*, vol. 34, no. 1, pp. 1–54, Apr. 1999. [Online]. Available: http://www.elsevier.com/cas/tree/store/scico/sub/1999/34/1/559.pdf

[19] E. Guerra and J. de Lara, "A framework for the verification of UML models. examples using petri nets," pp. 325–334, 2003.

[20] T. S. Staines, "Intuitive mapping of UML 2 activity diagrams into fundamental modeling concept petri net diagrams and colored petri nets," pp. 191–200, 2008. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ECBS.2008.12